Aggregate-Signcryption for Securing Smart Camera IoT Applications

Subhan Ullah*†, Federico Russo†, Lucio Marcenaro† and Bernhard Rinner*
*Institute of Networked and Embedded Systems
Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria
Email: {subhan.ullah, bernhard.rinner}@aau.at
†Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture
University of Genova, Via all'Opera Pia 11, 16145 Genova, Italy
Email: lucio.marcenaro@unige.it

***Abstract*—Smart cameras are considered as key sensors in Internet of Things (IoT) applications ranging from home to city scales. Since these cameras often capture highly sensitive information, security is a major concern. An elliptic curve (EC) based signcryption achieves resource-efficiency by performing data encryption and signing in a single step. In this work, we present aggregate-signcryption which extends the EC-based signcryption approach to a cluster-based multi-camera setup. The signcrypted data from the smart cameras within a cluster is aggregated on a specific node called cluster head. Aggregate-signcryption reduces the communication overhead and requires fewer steps for the unsigncryption as compared to individual signcryption.**

***Index Terms*—Smart cameras; Aggregate-signcryption; Data Security; Internet of Things;**

I. Introduction

Smart cameras are real-time computer vision systems that combine onboard sensing, processing and communication capabilities [1]. These devices play an important role in several IoT applications [2]. However, security and privacy protection has become a major concern due to their widespread deployment, the sensitive nature of the captured data and the open infrastructure [3]. Visual data captured by smart cameras might reveal not only the identity but also clues about the habits, preferences and social interests of a person [4]. The basic security objective for a smart camera is thus to prove the originality of images or video data (integrity), its origin (authenticity) and the protection from unauthorized access (confidentiality) throughout the entire lifetime of the data.

In our preliminary work [5], an EC-based signcryption technique has been proposed as an efficient solution for securing the smart camera by separating the platform into a trusted sensing unit and an untrusted camera host unit [6]. In that case, security was ensured by individually protecting images transmitted from the smart camera. In an IoT environment more than one smart camera are often required to monitor a wide area [7]. The transmission of signcrypted images or video frames from co-located distinct smart cameras at the same time might saturate the communication channel and overload the end-user device during the verification and unsigncryption process.

In this work, we propose a cluster-based multi-camera architecture (Figure 1) which is able to efficiently process and secure the captured data. We divide the network of smart cameras into distinct clusters and extend the EC-based signcryption [5] to an aggregate-signcryption approach. The aggregate-signcryption is entirely performed on a cluster head by merging the signcryptexts and corresponding public keys of the smart cameras in the same cluster. Aggregate-signcryption combines signcryptexts to reduce the signature data without losing any security properties of the individual signcryptexts. The aggregate-signcryption saves communication costs during transmission and computation resources during verification on the end-user device (e.g., smartphone). The clustering approach provides scalability and management to the network of smart cameras, where aggregate-signcryption provides an efficient approach for data protection.

The contribution of this work lies in the deployment and evaluation of EC-based aggregate-signcryption in cluster-based smart camera IoT applications. We enable the cluster head to efficiently apply aggregation on the collected signcryptexts of distinct smart cameras. The proposed architecture has been implemented and evaluated on a network based on Raspberry Pi nodes.

The rest of the paper is organized as follows: Section II discusses the state-of-the-art. Section III introduces the system architecture, assumptions and threat model. Sections IV and V describe the proposed solution and experimental evaluation, respectively. Finally, Section VI concludes the paper.

II. State-of-the-art

In the following, we briefly review smart camera based surveillance systems and security techniques for captured image and video data in the context of IoT [8] and visual sensor networks (VSN) [3]. The main focus of these applications is video surveillance of private [9] [10] or public premises [11].

A. Smart cameras in IoT applications

A smart camera is a key sensor for video surveillance in IoT applications. Najjar et al. [12] briefly introduced some basic VSN platforms (Cyclops, MeshEye, Vision Mote, MicrelEye), related architectures and challenges. They further highlighted

the need of lightweight algorithms for image processing and identified the trade-off between accuracy of algorithms, memory, processing and power consumption. Winkler and Rinner [10] presented a novel platform, TrustEYE.M4, to provide security and privacy of data in VSN applications. Baran et al. [13] used a smart camera for the identification and recognition of vehicles in transportation system for the law enforcement authorities.

B. Video surveillance systems

A generic architecture of video surveillance consists of smart cameras, backup servers and end-user monitoring devices [14] [9]. A network of smart cameras is required for the surveillance of a large area, and can be classified into centralized, distributed or cluster-based networks [15]. Natarajan et al. [16] summarized the related work of multi-camera on object detection, tracking, security, privacy, coordination and control strategies for video surveillance applications. Chien et al. [8] used a special node for aggregation of data from video sensors in the IoT-based video surveillance and recommended lightweight algorithms and system on chip (SoC) approaches to further improve the computation power of sensors. Mora et al. [17] proposed an IoT-based framework for healthcare monitoring and proposed a scheduling technique for sharing the resources among the nodes to minimize computational costs. This approach preserved local resources for critical processing only.

C. Security approaches in video surveillance

In video surveillance a smart camera captures large volumes of data in the form of images and videos and efficient security techniques are needed for data protection [3]. Usually, standard security techniques are used to provide confidentiality, authenticity and integrity of the data [18] in IoT applications. Alsmirat et al. [19] presented a framework for secure surveillance system [20]. They used the advanced encryption standard (AES) for confidentiality and the RSA algorithm for key distribution. The session key was further secured by using (HMAC-MD5) hashing and provided authentication and integrity of the video streams. This approach was implemented by using the NS-3 simulator for evaluating the trade-off between communication delay and security. The computation and communication overheads were reduced by encrypting the whole video frame instead of encrypting each packet of data. Winkler and Rinner [10] used the hardware-based trusted platform module (TPM) security chip for onboard security and privacy protection on the smart camera. They used AES for encryption and the RSA digital signature for signing with time-stamping techniques, and proved confidentiality, integrity and authentication for the captured data. Haider and Rinner [9] used on-chip physical unclonable functions (PUF) and provided onboard protection for image and video data. They followed the encrypt-then-sign approach by using PUF-based secure key generation. They evaluated the results on a Zynq7010 SoC-based prototype. A secure remote authentication of the user was performed for the smart cities applications [21]. A PUF-based CMOS image sensor was proposed by [22].

D. Comparison with our approach

We implemented the EC-based signcryption techniques [5] on each smart camera in a cluster-based network for securing the video frames and proposed a cluster head as an aggregator for all the signcryptexts to reduce the computation and communication overhead. The smaller key size of EC [23] and the implementation of signcryption [24] supports real-time data security directly on the sensing unit. To the best of our knowledge, our approach is the first deployment of aggregate-signcryption in this context.

III. System architecture

In this section, we present our proposed system architecture, its integral components and connectivity in a typical IoT environment (Figure 1). The integral components are smart cameras, a backup server and a set of monitoring devices. The smart cameras are able to detect predefined events due to their local processing capabilities. Once an event has been detected, the camera triggers a description of the event and identifies a region of interest. We group the co-located smart cameras into distinct clusters [25]. Each cluster has a pre-defined cluster head, which works as a gateway [26] and connects the smart cameras with the rest of the system. The identities of the cluster heads and its corresponding smart cameras are represented as CH_i and $C_{j.i}$ where i and j represent the identifiers for the cluster and the camera, respectively. Typically, a smart camera has not sufficient storage for all captured data because of resource limitations, so we use a backup server (BS_1) to permanently store the aggregate-signcrypted data forwarded by the cluster heads for the intended monitoring device. The backup server provides an authorized access to that stored data for the corresponding monitoring device (M_h).

Many-to-one-communication scenario: In this work, we extend our previous security approach [5] to a many-to-one communication scenario. Here, multiple cameras provide data for detected events and need to secure it for an individual monitoring device. The processing for such scenario can be summarized as follows: (i) onboard detection of predefined events on smart cameras in a cluster, (ii) aggregation of the information on the cluster head, (iii) transfer and storage of the aggregated information on a backup server, (iv) and download of the information by monitoring device which are already stored on the backup server to complete the surveillance procedure.

A. Assumptions

We assume that each smart camera consists of a trusted sensing unit and camera host unit [6]. The host unit is not explicitly trusted because of the operating system (OS), libraries, middle-ware and other user-specific applications. The camera host unit is responsible for the configuration, management and running of the application and system libraries. The protection of the sensing unit is built upon our previous work [10], [9]

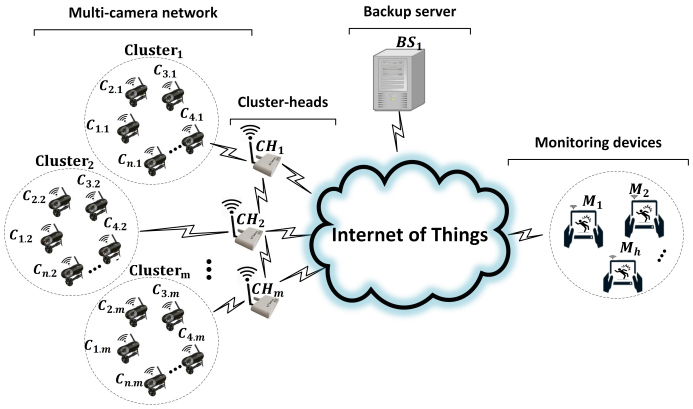

Fig. 1. The system architecture composed by several clusters of smart cameras, dedicated cluster heads, monitoring devices and a backup server.

and has exclusive access to the raw data (images and videos). We assume no explicit protection of the denial of service (DoS) attacks on the other components of the system. While assuming the sharing of public parameters and public keys, we can verify the authenticity and reduce incoming requests of an attacker by using public verification (a property of signcryption technique). Moreover, we assume that the monitoring device is trustworthy and that private keys are securely stored on it.

B. Threat model

In the proposed system architecture (Figure 1), an attacker may get access to image or video data by compromising the camera host unit, the cluster head or the backup server. The attacker can access the data possibly on the camera host part, cluster head, and communication channels or backup server, as reported in the context of IoT smart home [27] and VSN [3] scenarios. The attacker can compromise the integrity and alter the data while remain undetected. Another capability of the attacker is to compromise the authenticity and insert their own information by using the identity of smart cameras.

IV. Proposed solution

In this section, we present lightweight security techniques to protect and secure the sensitive information in the proposed system architecture. The design goals of the security techniques are (i) to reduce the transmission of unnecessary data, (ii) to protect the captured information from unauthorized access throughout its lifetime, and (iii) to prove the authentication and integrity of the information on the intended monitoring devices. In the following, we present an overview of the lightweight security approaches, as well as its deployment, and operational phases in the system architecture.

A. Overview of security techniques

We present aggregate-signcryption, in order to merge the signcrypted information from individual smart cameras of a cluster and to reduce the transmission of redundant information. The deployment of aggregate-signcryption requires the following steps:

Key generation: We use a key generation center (KGC) [28], a trusted entity in the system to generate the partial private keys for all devices of the system. The KGC securely shares the partial private keys with the respective devices, and then the devices generate their full private and public keys. We assume that all devices keep the private keys secret and share their public keys with each other in the system.

Local analysis and onboard signcryption: The smart cameras perform local event detection and then extract the region of interest (RoI). The smart camera applies EC-based signcryption to protect the selected data on the sensing unit [5]. The protected information is forwarded to the camera host unit, which can verify the integrity of the data and transfers it to the cluster head for aggregate-signcryption.

Aggregate-signcryption: The cluster head of each cluster verifies and aggregates the individual signcryptexts of the received data from the detected event. The algorithm of aggregate-signcryption is defined in Section IV-C. There is no need to share private keys with the cluster head but only the identities and public keys of the corresponding smart cameras and the receiving monitoring device is needed as input for the aggregate-signcryption. An aggregate-signcryption efficiently merges the signcryptexts of distinct smart cameras into a single and smaller aggregate-signcryptext. It merges the public key information of the intended monitoring device in a compact form. Then the monitoring device uses that information and verifies the authenticity and integrity of all data in a single step. Aggregate-signcryption does not affect the security of individual signcryptexts.

Permanent backup of data and accesses authorization: The cluster head forwards the aggregate-signcryptexts to a backup server for permanent storage and alerts the monitoring device. The monitoring device accesses the relevant aggregated-signcryptext and then performs the verification and unsigncryption on it.

B. Deployment phase

In deployment phase, the system initiates a setup of the entities and shares the identities along with the associated public keys and other state information.

Setup initialization: The KGC runs the setup algorithm and takes $k \in \mathbb{Z}^+$ as input (k specifies the bit length) to generates the partial private keys and public parameters [28] [24]. An EC over the finite field F_p is represented by $E(F_p)$ with a base point $G \in F_p$ of order q, where G is chosen randomly from the set of points on $E(F_p)$. The parameter p is a prime number specifying the finite field F_p. It is assumed that each device generates its full private key sk and the public key pk on the basis of the partial private key. The smart cameras $C_{j.i}$ generate their private keys as $sk_{C_{j.i}}$ and the public keys $pk_{C_{j.i}}$. The monitoring device also generates its private key sk_{M_1} and the public key pk_{M_1}. Each device keeps the private key secret and shares the public key during the initialization of the system or joining of a new device.

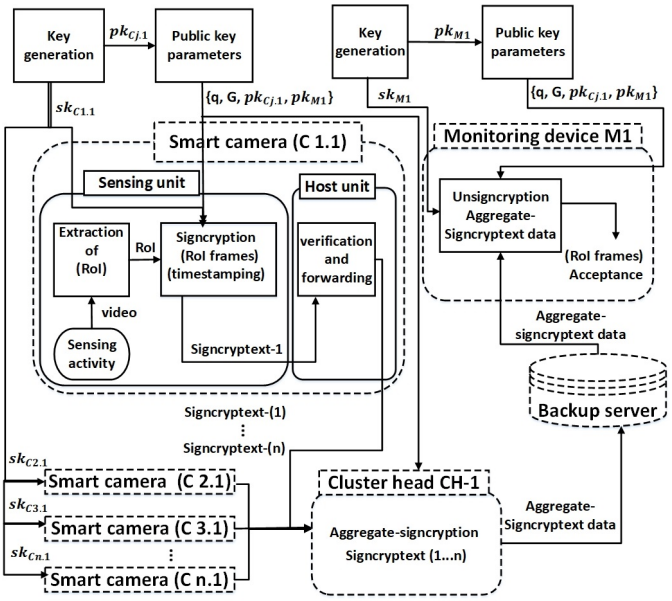

Fig. 2. Processing flow of the cluster-based aggregate-signcryption.

C. Operational phase

In the operational phase each smart camera initiates the signcryption process and generates a session key $k_{2(j)}$ by using the public key pk_{M_1} of a monitoring device. The smart camera uses its private key $sk_{C_{j.i}}$ for the signature part, while the session key for the encryption part and performs signcryption on the captured data as following.

Signcryption by smart camera: Let's suppose that a smart camera of cluster i detects an event and starts the signcryption procedure. Each smart camera $C_{j.i}$ selects the internal state information ω (e.g., number of smart cameras in the cluster, system time) and then performs the signcryption by executing the following steps:

- selection of a prime number $v_j \in \mathbb{Z}_q^*$,
- computation of $k_{1(j)} = hash(v_j.G)$,
- generation of the session key as $k_{2(j)} = hash(v_j.pk_{M_1})$,
- encryption of the RoI of the video frames.

$$c_j = enc_{k_{2(j)}}(RoI_{frames_{(timestamps)}})_j \quad (1)$$

$$r_j = hash(c_j, k_{1(j)}) \quad (2)$$

$$s_j = \frac{v_j}{(r_j + sk_{c_{j.i}})} \, mod \, q \quad (3)$$

$$R_j = (r_j.G) \quad (4)$$

$$Signcryptext = (c_j, R_j, s_j) \quad (5)$$

Each smart camera forwards its signcryptext packet (c_j, R_j, s_j) to the cluster head i.

Aggregate-signcryption algorithm: The cluster head i performs the aggregation of the individual signcryptexts received from the cameras. The aggregate-signcryption takes the public keys of the smart cameras $pk_{C_{j.i}}$, the public key of monitoring device pk_{M_1} and corresponding signcryptexts. The cluster head first verifies the individual signcryptexts and then generates the aggregate-signcryptext as following:

- computing $S = \sum_{j=1}^{n} s_j$ and parse the c_j and R_j in a specific order.
- merging signatures and encrypted data $(c_j \cdots c_n, R_j \cdots R_n, S)$ as aggregate-signcryptext.

Aggregate unsigncryption algorithm: prior to the decryption of aggregated data, the monitoring device first verifies the acceptance (authentication and integrity) of the aggregate-signcryptext data by using its own private key sk_{M_1}, the associated public keys of the smart cameras $pk_{C_{j.i}}$ and the received aggregate.signcryptext. In case of success, the output of the unsigncryption algorithm is the individual signcryptexts (c_j, R_j, s_j). This single step verification of aggregated data is true for all individual signcryptexts and there is no need to run the acceptance procedure individually. The monitoring device needs the individual session keys of the smart cameras to proceed with the decryption of c_j. It starts recovering the session keys $k_{2(j)} = hash(sk_{M_1}(s_j(R_j + pk_{C_{j.i}})))$ and then performs the decryption to get the required video frames of the RoI, e.g., $(RoI_{frames_{(timestamps)}})_j = dec_{k_{2(j)}}(c_j)$.
Figure 2 shows the processing flow of the signcryption, aggregation and unsigncryption procedure for many-to-one communication scenario. The correctness of the scheme to recover $k_{2(j)}$ on the motoring device is based on the following reasoning:
$hash(sk_{M_1}(s_j(R_j + pk_{C_{j.i}}))) = hash(sk_{M_1}(s_j.R_j + s_j.pk_{C_{j.i}})) = hash(sk_{M_1}(v_j.G)) = hash(v_j(sk_{M_1}.G)) = hash(v_j.pk_{M_1}) = k_{2(j)}$

D. Security analysis

The security analysis of the aggregate-signcryption scheme with specific attention to the system architecture can be summarized as follows: The basic security goals are confidentiality, integrity, authenticity and freshness of the image or video data. The security of signcryption is based on the assumption of computational hardness of EC-based discrete logarithm problem (ECDLP) [29].

Confidentiality: Confidentiality is provided by AES encryption using a session key $k_{2(j)}$ during the signcryption process. The guessing of $k_{2(j)}$ by attackers corresponds to solving the ECDLP.

Integrity: The sensing unit of smart camera processes a valid signcryption part r_j by hashing the encrypted data c_j with $k_{1(j)}$ as in Eq. (2). If an attacker modifies the encrypted data c_j to c_j', the change will be detected on the monitoring device because of collision resistance of the hash function.

Authentication: The signcryption technique provides the authentication and prove the authenticity of data e.g.,
if $hash(s_j(R_j + pk_{C_{j.i}})) = hash(v_j.G) = (k_{1(j)})$.
The correctness of the scheme to recover $k_{1(j)}$ on the monitoring device is based on the following reasoning,
$s_j(R_j + pk_{C_{j.i}}) = s_j.R_j + s_j.pk_{C_{j.i}} = (\frac{v_j}{(r_j+sk_{c_{j.i}})})R + (\frac{v_j}{(r_j+sk_{c_{j.i}})})pk_{C_{j.i}} = (\frac{v_j}{(r_j+sk_{c_{j.i}})})r_j.G + (\frac{v_j}{(r_j+sk_{c_{j.i}})})sk_{C_{j.i}}.G = \frac{v_j.G(r_j+sk_{c_{j.i}})}{(r_j+sk_{c_{j.i}})} = v_j.G = k_{1(j)}$

Freshness of the captured data: Image or video frames are timestamped during signcryption and the monitoring device verifies the validity after the processing of unsigncryption.

V. Experimental evaluation

In this section, we compare the computational and communication overhead for individual and aggregate-signcryption. A complete prototype of our system architecture (Figure 1) has been implemented and can be summarized as follows: Raspberry Pi 3 serve as platforms for the smart cameras. We use the JRPiCam [30] Java library for image capturing and processing. Each image has a pre-defined QVGA resolution of 320×240 pixels. The open source library BouncyCastle [31] is used as cryptographic service provider (CSP) with the Java cryptography extension (JCE) and the Java cryptography architecture (JCA) as interface. Signcryption is implemented using the EC-finite field of P-384 and a 256 bit AES key. The cluster head is implemented on a standard laptop (core i5 with 2.6 GHz and 8 GB RAM) running Windows 10. We used another laptop as prototype for the monitoring device. All platforms are connected via WiFi and data transfer is realized via sockets.

A. Experimental results

In the first experiment, we measured and evaluated the computational and communication overheads of individual-signcryption and aggregate-signcryption by securing 15 images (total size of 74.854 kB data) on each Raspberry Pi device. First, we varied the number of devices for initiating the signcryption at the same time and measured the total computation and communication overhead of signcryption (on Raspberry Pi 3) and unsigncryption (on the monitoring device). Second, we measured the total computation and communication overhead for aggregate-signcryption (on Raspberry Pi 3 and on the laptop used as cluster head) and aggregate-unsigncryption (on another laptop used as monitoring device). Table I shows the comparison of individual and aggregate-signcryption with varying number of smart cameras.

In the second experiment, we varied the number of images and thus the data size and measured the signcryption time (on Raspberry Pi 3) and unsigncryption time (on monitoring device). We performed this experiment in a cluster of five cameras where each camera secured a different number of images. Table II shows the measured runtimes for signcryption and unsigncryption, respectively. The total time for individual-signcryption can be determined for the second experiment as follows: Signcryption is executed in parallel on the cameras, thus the maximum runtime (760 ms) is the limiting factor for this step. Unsigncryption has to be performed sequentially on the monitoring device and can be estimated by the sum of the unsigncryption times (1502 ms) resulting in a total time of 2262 ms. For aggregate-signcryption, the total time is given by the maximum signcryption time (760 ms), the aggregate-signcryption time (349 ms) and the aggregate unsigncryption time (634 ms) which sums up to 1743 ms resulting in a performance ratio of 77%. Table II also shows that the signcryption time only slightly increases with increasing data size. This effect is because the intensive EC-point computations needs to be executed at the beginning of the signcryption process and only the encryption algorithm is dependent on the data size.

As depicted in Table I, aggregate-signcryption shows a moderate increase of the runtime with increasing number of cluster cameras. This additional effort of aggregate-signcryption is clearly compensated by the signification reduction of unsigncryption time, in particular with larger numbers of cluster cameras. Table I also shows that the ciphertext part c_j of each signcryptext packet (c_j, R_j, s_j) has the same size of 74.854 kB (size of 15 images) for the individual-signcryption and aggregate-signcryption, while the signature part varies with the number of cameras in the cluster. As we use the finite field P-384, so the signature part (s_j) in Eq. (3) of signcryption scheme results in 48 Bytes. The r_j in Eq. (2) has 48 Bytes because of the keyed hash function of SHA-384 and it is further used for the computation of the R_j part using the finite field P-384 as shown in Eq. (4) of the signcryption algorithm which also results in 48 Bytes. However, we use point compression [32] for R_j part which reduces the size to the half of its length (24 Bytes). Hence the total extra overhead per individual signcryptext of R_j and s_j results in 72 Bytes. In the case of individual signcryption each signcryptext carry 72 Bytes of extra data, while in the case of aggregate-signcryption the s_j part is merged into S using the finite field of P-384, which results in 48 Bytes for the aggregated packet and decreases the extra communication overhead.

VI. Conclusion

In this work, we investigated the performance of aggregate-signcryption for cluster-based smart camera IoT applications. First, we implemented the EC-based signcryption for the security of multiple images on smart camera and reduced the average running time per image. Second, we implemented the aggregate-signcryption and investigated the performance in cluster-based multi-camera network. We reduced the communication and computation overheads by implementing aggregate-signcryption. We evaluated the performance ratio between individual and aggregate-signcryption for communication and computation overhead in many-to-one communication scenario. In future, we will extend this work for one-to-many and many-to-many communication scenarios.

Acknowledgment

This work is supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments, which is funded by the Education, Audiovisual & Culture Executive Agency. This work is also supported by the research initiative Mobile Vision Austria with funding from the Austrian Federal Ministry of Science, Research and Economy and the Austrian Institute of Technology.

References

[1] W. Wolf, B. Ozer, and T. Lv, "Smart cameras as embedded systems," *Computer*, vol. 35, no. 9, pp. 48–53, Sep 2002.

TABLE I

COMPARISON BETWEEN INDIVIDUAL SIGNCRYPTION VS AGGREGATE-SIGNCRYPTION. LEGENDS: (ST: SIGNCRYPTION TIME (ON CAMERA NODES), UST: UNSIGNCRYPTION TIME (ON MONITORING DEVICE), TT: TOTAL TIME, NT: NUMBER OF TRANSFERS, CD: CIPHERTEXT DATA, SD: SIGNATURE DATA, AST: AGGREGATE SIGNCRYPTION TIME (ON CLUSTER HEAD), AUST: AGGREGATE UNSIGNCRYPTION TIME (ON MONITORING DEVICE))

		Individual-signcryption						**Aggregate-signcryption**						**Performance ratio**	
		ST(ms)	UST (ms)	TT (ms)	NT	CD (kB)	SD (Bytes)	ST (ms)	AST (ms)	AUST (ms)	TT (ms)	NT	SD (Bytes)	Computing	Signature data
No. cluster members	1	741	358	1099	1	74.854	72	741	138	358	1237	2	72	112.5%	100%
	2	741	716	1457	2	149.708	144	741	264	388	1393	3	96	95.6%	66.6%
	3	741	1074	1815	3	224.562	216	741	280	453	1474	4	120	81.2%	55.5%
	4	741	1432	2173	4	299.416	288	741	325	581	1647	5	144	75.8%	50%
	5	741	1790	2531	5	374.270	360	741	389	700	1830	6	168	72.3%	46.6%

TABLE II

RUNTIME OF SIGNCRYPTION (ON CAMERA NODE) AND UNSIGNCRYPTION (ON MONITORING DEVICE) FOR DIFFERENT NUMBER OF IMAGES.

Camera ID	**No. images (size in kB)**	**Signcryption time (ms)**	**Unsigncryption time (ms)**
1	1 (5.173)	636	221
2	5 (25.077)	677	242
3	10 (49.99)	721	313
4	15 (74.854)	741	358
5	20 (99.717)	760	368

[2] M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Smart camera networks," *Computer*, vol. 47, no. 5, pp. 23–25, May 2014.

[3] T. Winkler and B. Rinner, "Security and privacy protection in visual sensor networks: A survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 2:1–2:42, May 2014.

[4] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 636–654.

[5] S. Ullah, B. Rinner, and L. Marcenaro, "Smart cameras with onboard signcryption for securing iot applications," in *Proc. IEEE Global Internet of Things Summit (GIoTS)*, June 2017, pp. 1–6.

[6] T. Winkler, A. Erdelyi, and B. Rinner, "Trusteye.m4: Protecting the sensor not the camera," in *Proc. 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2014, pp. 159–164.

[7] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 187–198, Jan 2018.

[8] S. Chien, W. Chan, Y. Tseng, C. Lee, V. Somayazulu, and Y. Chen, "Distributed computing in iot, system-on-a-chip for smart cameras as an example," in *Proc. 20th Asia and South Pacific Design Automation Conference*, Jan 2015, pp. 130–135.

[9] I. Haider and B. Rinner, "Private space monitoring with soc-based smart cameras," in *Proc. IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Oct 2017, pp. 19–27.

[10] T. Winkler and B. Rinner, "Secure embedded visual sensing in end-user applications with TrustEYE.M4," in *Proc. IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Apr 2015, pp. 1–6.

[11] D. Lee and N. Park, "Geocasting-based synchronization of almanac on the maritime cloud for distributed smart surveillance," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 1103–1118, Mar 2017.

[12] M. Al Najjar, M. Ghantous, and M. Bayoumi, *Visual Sensor Nodes*. Springer, New York, 2014, pp. 17–35.

[13] R. Baran, T. Rusc, and P. Fornalski, "A smart camera for the surveillance of vehicles in intelligent transportation systems," *Multimedia Tools Appl.*, vol. 75, no. 17, pp. 10 471–10 493, Sep. 2016.

[14] F. Porikli, F. Bremond, S. L. Dockstader, J. Ferryman, A. Hoogs, B. C. Lovell, S. Pankanti, B. Rinner, P. Tu, and P. L. Venetianer, "Video surveillance: past, present, and now the future [dsp forum]," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 190–198, May 2013.

[15] H. Aghajan and A. Cavallaro, *Multi-Camera Networks: Principles and Applications*. Academic Press, 2009.

[16] P. Natarajan, P. K. Atrey, and M. Kankanhalli, "Multi-camera coordination and control in surveillance systems: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 4, pp. 57:1–57:30, Jun. 2015.

[17] H. Mora, D. Gil, R. M. Terol, J. Azorn, and J. Szymanski, "An iot-based computational framework for healthcare monitoring in mobile environments," *Sensors*, vol. 17, no. 10, 2017.

[18] J. H. Kong, L.-M. Ang, and K. P. Seng, "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments," *Journal of Network and Computer Applications*, vol. 49, no. Supplement C, pp. 15 – 50, 2015.

[19] M. A. Alsmirat, I. Obaidat, Y. Jararweh, and M. Al-Saleh, "A security framework for cloud-based video surveillance system," *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22 787–22 802, Nov 2017.

[20] M. A. Alsmirat, Y. Jararweh, I. Obaidat, and B. B. Gupta, "Internet of surveillance: a cloud supported large-scale wireless surveillance system," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 973–992, Mar 2017.

[21] G. Sharma and S. Kalra, "A secure remote user authentication scheme for smart cities e-governance applications," *Journal of Reliable Intelligent Environments*, vol. 3, no. 3, pp. 177–188, Sep 2017.

[22] Y. Cao, L. Zhang, S. S. Zalivaka, C. H. Chang, and S. Chen, "Cmos image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 11, pp. 2629–2640, Nov 2015.

[23] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of cryptology*, vol. 14, no. 4, pp. 255–293, 2001.

[24] E. Mohamed and H. Elkamchouchi, "Elliptic curve signcryption with encrypted message authentication and forward secrecy," *International Journal of Computer Science and Network Security*, vol. 9, no. 1, pp. 395–398, 2009.

[25] A. A. Zarezadeh, C. Bobda, F. Yonga, and M. Mefenza, "Efficient network clustering for traffic reduction in embedded smart camera networks," *Journal of Real-Time Image Processing*, vol. 12, no. 4, pp. 813–826, Dec 2016.

[26] V. P. Venkatesan, C. P. Devi, and M. Sivaranjani, "Design of a smart gateway solution based on the exploration of specific challenges in iot," in *Proc. International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC)*, Feb 2017, pp. 22–31.

[27] J. Pacheco and S. Hariri, "Iot security framework for smart cyber infrastructures," in *Proc. International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Sep 2016, pp. 242–247.

[28] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology - ASIACRYPT 2003*, C.-S. Laih, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 452–473.

[29] M. Yasuda, T. Shimoyama, J. Kogure, and T. Izu, "Computational hardness of ifp and ecdlp," *Applicable Algebra in Engineering, Communication and Computing*, vol. 27, no. 6, pp. 493–521, 2016.

[30] A. Dillon, https://github.com/Hopding/JRPiCam, [Last accessed: 15-04-2018].

[31] https://www.bouncycastle.org/, [Last accessed: 15-04-2018].

[32] M. Khabbazian, T. A. Gulliver, and V. K. Bhargava, "Double point compression with applications to speeding up random point multiplication," *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 305–313, March 2007.